Top-to-Bottom, End-to-End Performance Monitoring for Applications of The Distributed-Parallel Storage System in the MAGIC Testbed[1]

Brian Tierney (bltierney@lbl.gov),
William Johnston(wejohnston@lbl.gov),
Gary Hoo, Guojun Jin, Jason Lee
Imaging and Distributed Computing Group
(http://www-itg.lbl.gov)
Lawrence Berkeley National Laboratory
Berkeley, CA 94720

1. **This work is jointly supported by ARPA - CSTO, and by the U. S. Dept. of Energy, Energy Research Division, Mathematical, Information, and Computational Sciences office, under contract DE-AC03-76SF00098 with the University of California. This document is report LBL-38396**

The MAGIC Testbed

- **The MAGIC testbed consists of a large-scale, high-speed, ATM network. It is a heterogeneous collection of:**
  - **ATM switches and computing platforms**
  - **IP over ATM implementations**
  - **"middleware" (distributed services), etc.,**

  all of which must cooperate in order to make complex applications operate at high speed

- **TerraVision (a virtual reality-like, terrain navigation application) and the Distributed-Parallel Storage System (DPSS - supplies data in real time to support interactive navigation of very large data sets) together are the model, high performance distributed application in MAGIC.**

MAGIC Testbed

- **The performance monitoring work is a cooperative effort of the MAGIC participants:**
  - **Earth Resources Observation System Data Center, U.S. Geological Survey (EDC)**
  - **Lawrence Berkeley National Laboratory, U.S. Department of Energy (LBNL)**
  - **Minnesota Supercomputer Center, Inc. (MSCI)**
  - **SRI International (SRI)**
  - **University of Kansas (KU)**
  - **CNRI/MITRE Corporation (project coordination)**
  - **Sprint**
  - **U S WEST Communications, Inc.**

Motivation and Approach

- **When building high-speed network-based distributed services, we often observe unexpectedly low network throughput and/or high latency - the reasons for which are usually not obvious**

  The bottlenecks can (and have been) in any of the components:

  - **the applications**
  - **the operating systems**
  - **the device drivers, the network adapters on either the sending or receiving host (or both)**
  - **the network switches and routers, and so on**

Motivation and Approach

- **It is difficult to track down performance problem because of the complex interaction between the many distributed system components resulting in problems in one place being most apparent somewhere else.**
- **Tools such at *ttcp* and *netperf* are somewhat useful but don't model real distributed applications, which are complex, bursty, and have more than one connection in and/or out of a given host at one time**

Motivation and Approach

- **We have developed a methodology and tools for monitoring, under realistic operating conditions, the behavior of all the elements of the application-to-application communications path in order to determine exactly what is happening within this complex system. We have:**
  - **instrumented our applications to do timestamping and logging at every critical point through the whole data path**
  - **modified some Unix network and operating system monitoring tools to log “interesting” events using a common log format**
  - **incorporate information from SNMP queries of the network components**

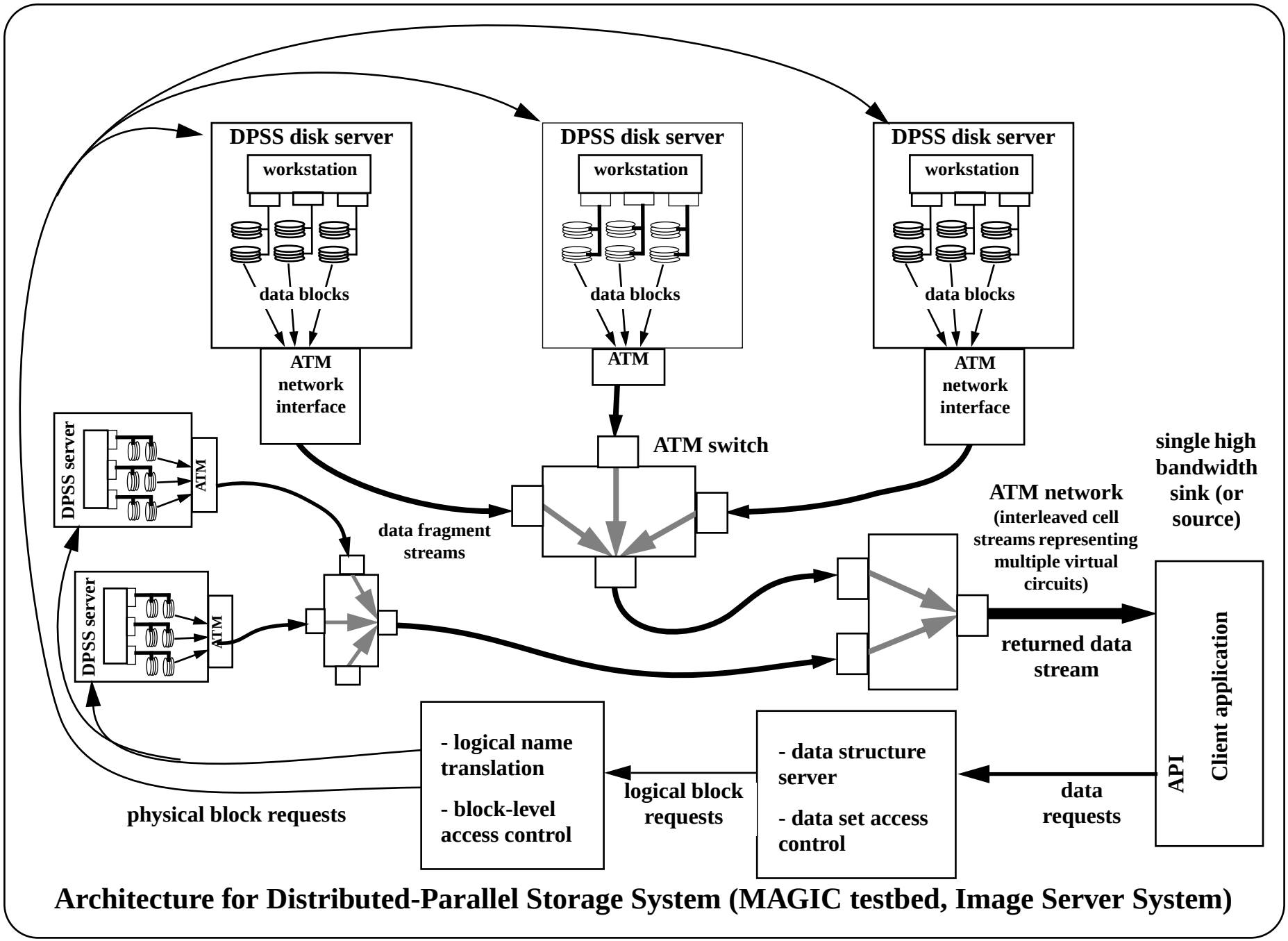
Motivation and Approach

- **The overall goal of this work is to identify what must be done to produce predictable, high-speed components that can be used as building blocks for high-performance applications, rather than having to "tune" the applications top-to-bottom as is all too common today.**

DPSS Architecture, Implementation, and Use

- **The Distributed-Parallel Storage System (DPSS, aka ISS) is a dynamically configurable network-striped disk array designed to supply high speed data streams to other processes in the network**
- **DPSS uses parallel operation of distributed servers to supply image streams fast enough to enable various multi-user, "real-time", virtual reality-like applications in an Internet / ATM environment**
- **At the application level, the DPSS is a persistent cache of named objects, at the storage level it is a logical block server (It is not a reliable tertiary storage system.)**
- **DPSS supports both read and write:**

DPSS



Architecture for Distributed-Parallel Storage System (MAGIC testbed, Image Server System)

DPSS

A medical imaging application in BAGNet currently stores 40 GBy/day from a SF hospital to a DPSS at LBNL.

This data is subsequently (in fact, potentially simultaneously) read by a display and analysis application.

DPSS

- **Client use of the DPSS**
  - **the client API is through two levels of library interface:**
    - **an object interface that presents the application view of the data**
    - **a DPSS interface that:**
      - **establishes authentication and authorization (X.509 certificates are used to establish a GSS security context)**
      - **sets up the “third party” data transfer connections to all the disk servers that hold pieces of the data set**
      - **provides the data block-request functions**
  - **data request prediction is an (essential) application function**

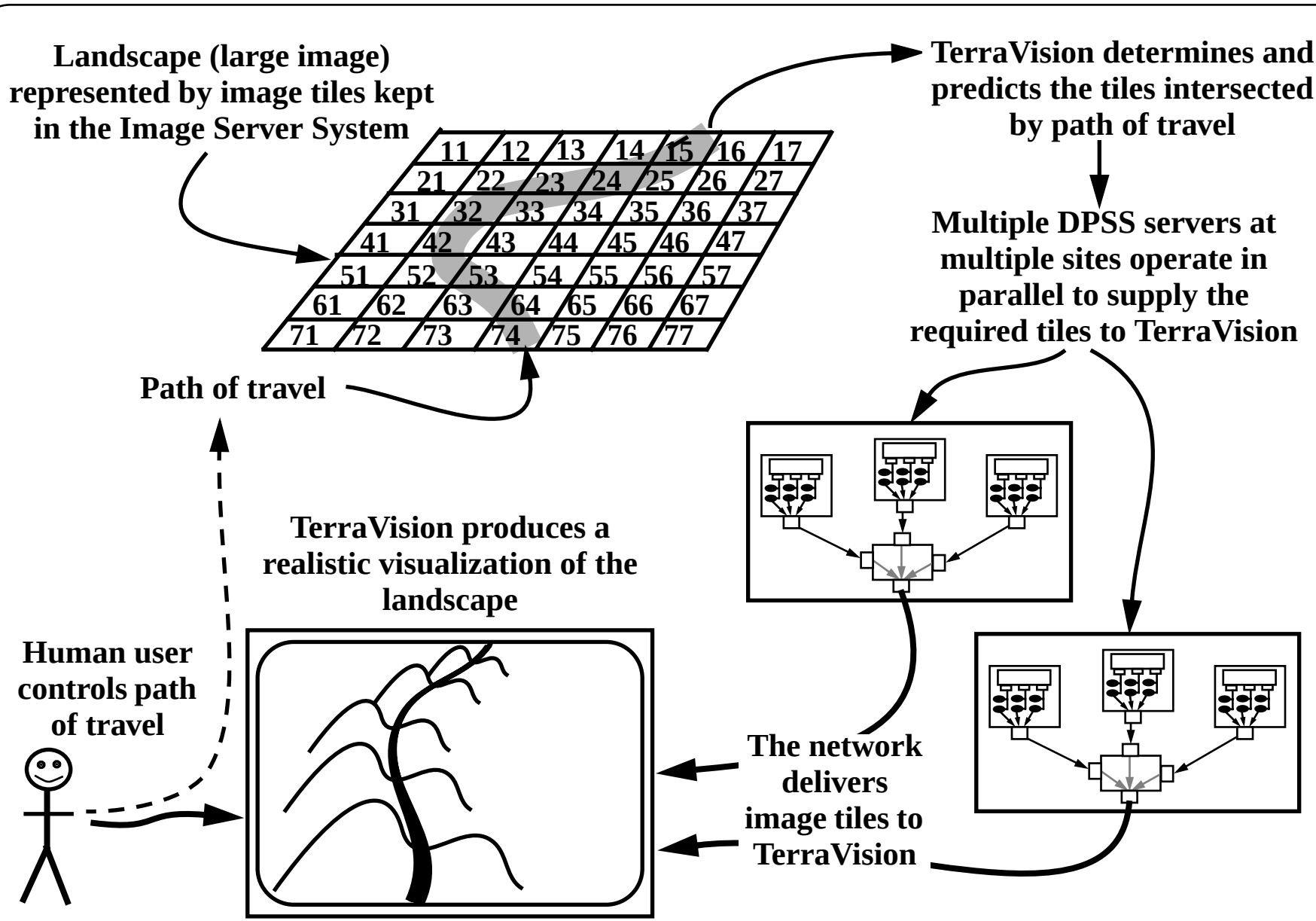
DPSS

- **Typical DPSS implementation**
  - **4 - 5 UNIX workstations (e.g. Sun SPARCStation, DEC Alpha, SGI, etc.)**
  - **4 - 6 fast-SCSI disks on multiple 2 - 3 SCSI host adaptors**
  - **an ATM network interface**
- **this configuration can deliver an aggregated data stream to an application at about 400 Mbits/s (50 MBy/s) using these relatively low-cost, "off the shelf" components by exploiting the parallelism provided by approximately five hosts, twenty disks, ten SCSI host adaptors, and five network interfaces**
- **transport is by either TCP or UDP/RTP, but performance testing is currently focused on TCP transport**

DPSS

- **TerraVision uses tiles images and digital elevation models to produce a 3D visualization of landscape.**
  - **the tiles are distributed across DPSS servers that are scattered around the MAGIC network**
  - **multiple DPSS servers operate in parallel to satisfy TerraVision data requests**

DPSS

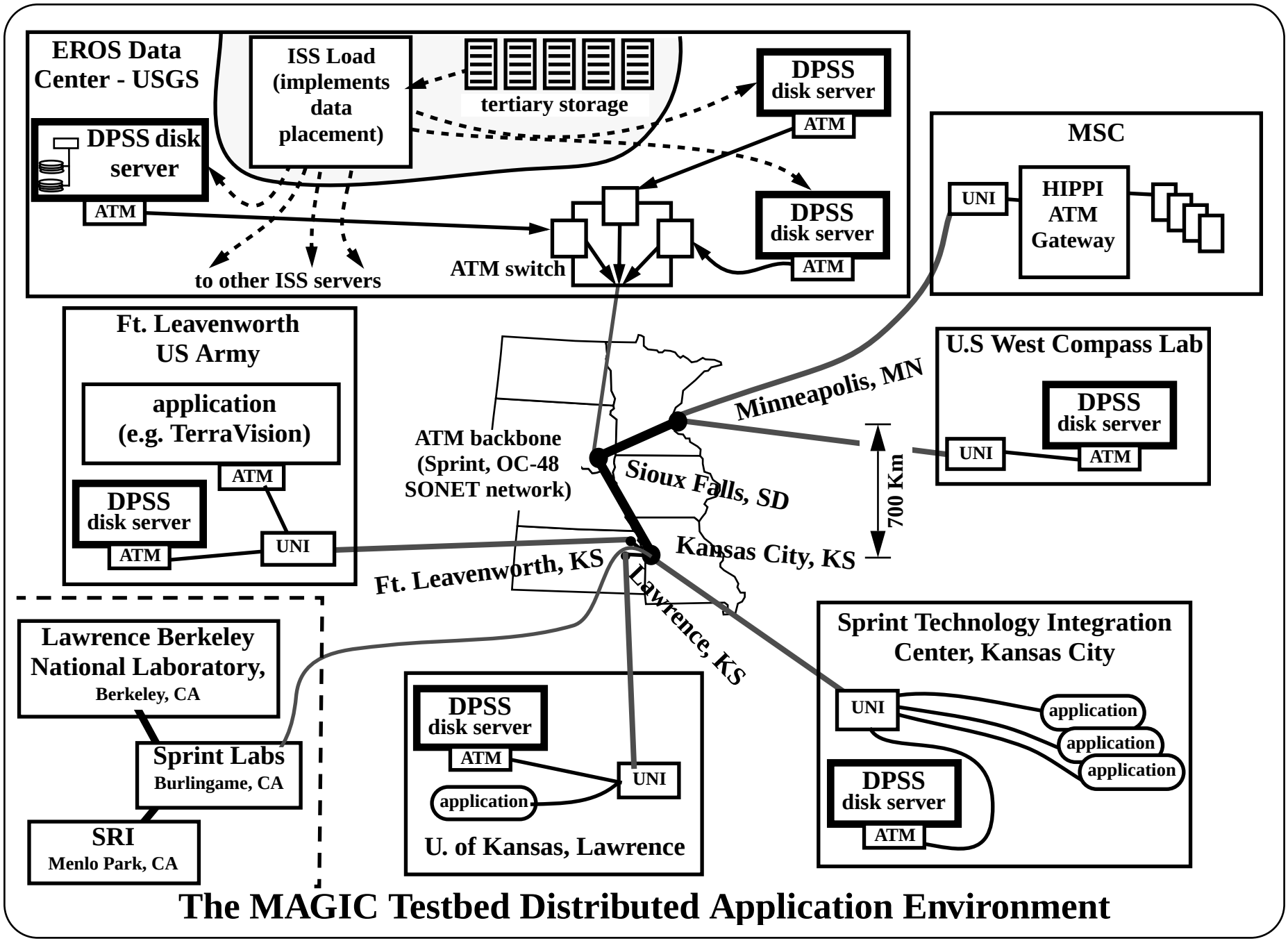


TerraVision and the DPSS Cooperate to Visualize Landscape

Performance Monitoring Approach and Experiments

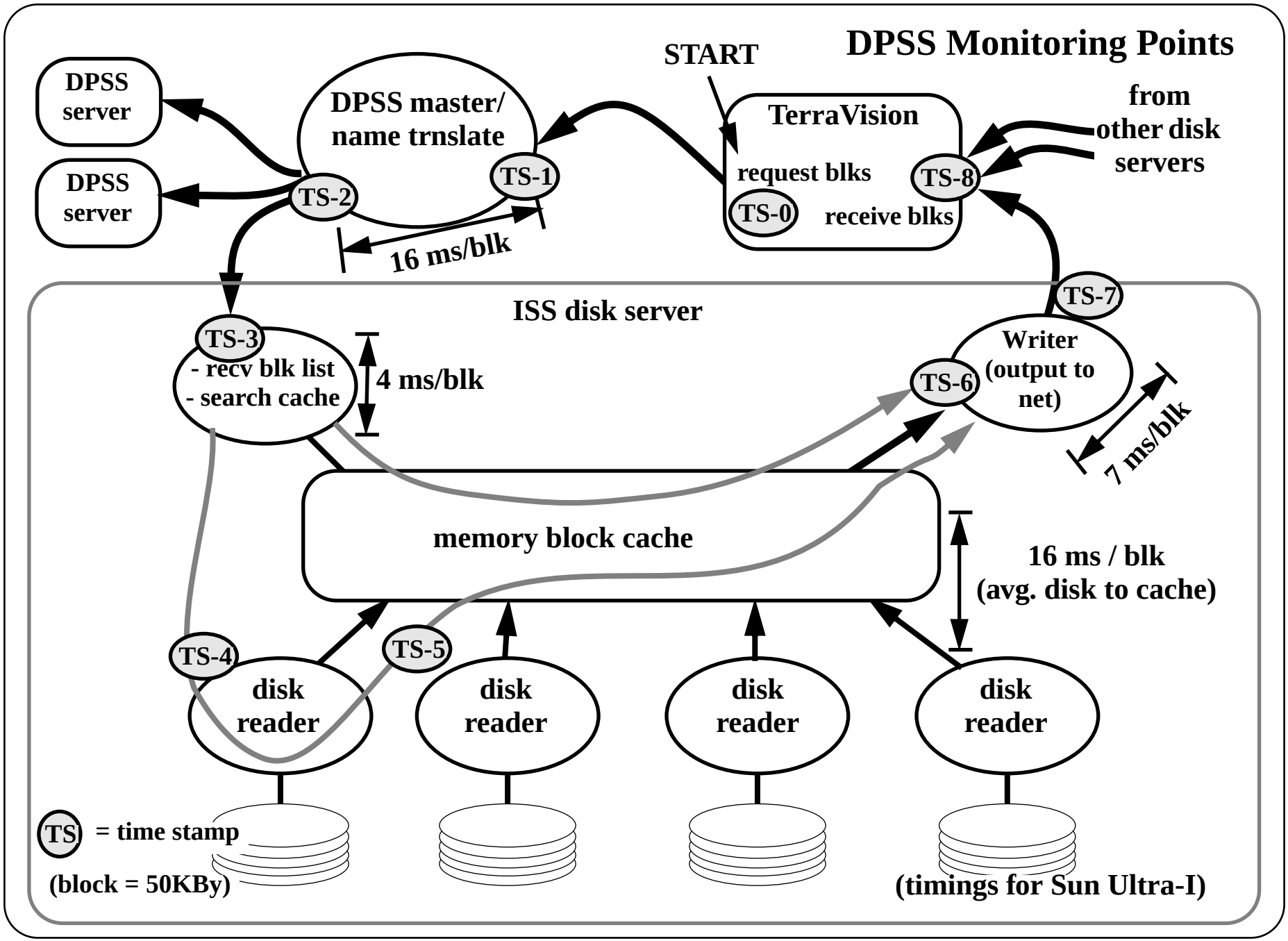
- **Performance monitoring in the application and in the DPSS involves collecting time stamps at all of the critical points in the data path**
- **Time and other monitoring information are made available to the DPSS master through monitoring protocols, and by being carried as a defined part of the data block structure**
- **At least three, and usually more, systems are typically involved in the monitoring experiments**

Monitoring



The MAGIC Testbed Distributed Application Environment

Monitoring


DPSS Monitoring Points
START
DPSS server
DPSS server
DPSS master/ name trnslate
TS-1
TS-2
16 ms/blk
TerraVision
request blks
TS-0
receive blks
TS-8
from other disk servers
ISS disk server
TS-3
- recv blk list
- search cache
4 ms/blk
TS-7
Writer (output to net)
TS-6
7 ms/blk
memory block cache
16 ms / blk
(avg. disk to cache)
TS-4
TS-5
disk reader
disk reader
disk reader
disk reader
TS = time stamp
(block = 50KBy)
(timings for Sun Ultra-I)

Monitoring

- **OS and network monitoring include:**
  - **TCP retransmits**
  - **CPU usage (user and system)**
  - **CPU interrupts**
  - **AAL 5 information**
  - **ATM switch buffer overflows**
  - **ATM hosts adapter buffer overflow**
- **Tools**
  - ***netstat* and *vmstat* modified to poll and report continuously**
    - **(Currently poll at 100 ms, so this data is accurate to +/- 50 ms)**
  - **SNMP queries to switches and network interfaces**
    - **switch buffer overflows (!)**

Monitoring

- **Common logging format:**

  keyword; hostname; seconds; nano-sec; data; data; data;.....;

  - **“keyword” - an identifier describing what is being logged - e.g. a reference to the program that is doing the logging: DPSS_SERV_IN, VMSTAT_SYS_CALLS, NETSTAT_RETRANSSEGS, TV_RQ_TILE**
  - **“data” elements (any number) are used to store information about the logged event - for example:**
    - **NETSTAT_RETRANSSEGS events have one data element: the number of TCP retransmits since the previous event**
    - **DPSS_START_WRITE events have data elements containing: the logical block name, the data set ID, a “user session” ID, and an internal DPSS block counter**

Monitoring

- **Experiment log records are “associated” by virtue of being collected and carried in the data block request message as it works its way through the system, and the request message is attached to the returned data block**

Monitoring

NETSTAT_OUTDATASEGS; 806706009; 652917; 4817378;
NETSTAT_RETRANSSEGS; 806706009; 652917; 16395;
NETSTAT_INUNORDERSEGS; 806706009; 652917; 797;
NETSTAT_RTTUPDATE; 806706009; 652917; 1762358;
NETSTAT_RETRANSBYTES; 806706009; 652917; 111841701;
NETSTAT_INUNORDERBYTES; 806706009; 652917; 547277;
NETSTAT_OUTDATABYTES; 806706009; 652917; 484913397;
NETSTAT_OUTDATASEGS; 806706009; 760199; 0;

VMSTAT_INTR; 806706009; 546568; 71612733;
VMSTAT_SYS_CALLS; 806706009; 546568; 2794466576;
VMSTAT_CONTEX_SW; 806706009; 546568; 2777829703;
VMSTAT_USER_TIME; 806706009; 546568; 2;
VMSTAT_SYS_TIME; 806706009; 546568; 5;
VMSTAT_INTR; 806706009; 646444; 3;

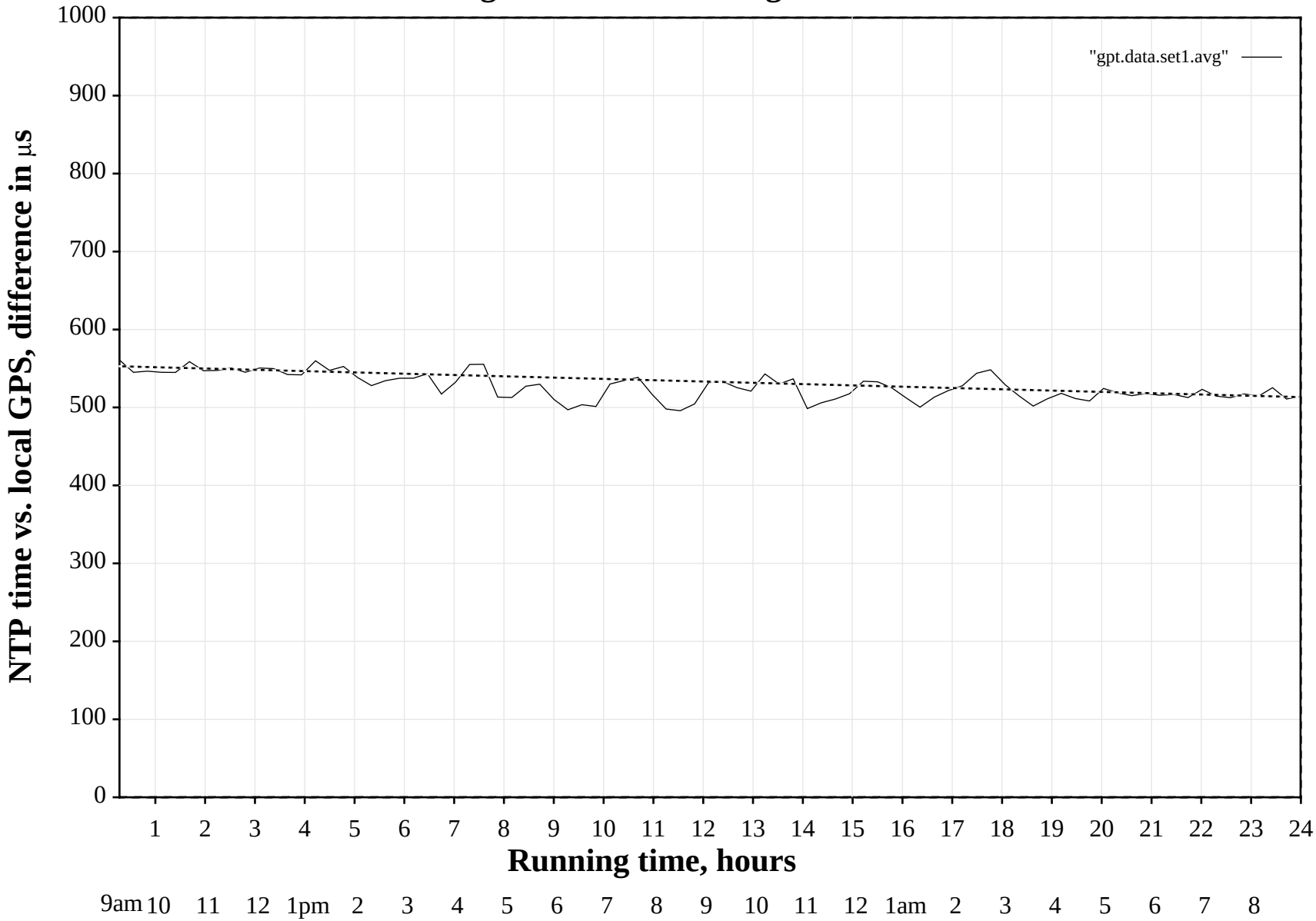
Monitoring

APP_SENT; 824951202; 824949; 34; 78; 45; 0; 6; 5; 198.207.141.6;
ISS_MASTER_IN; 824951202; 832232; 34; 78; 45; 0; 6; 5; 198.207.141.6;
ISS_MASTER_OUT; 824951202; 865724; 34; 78; 45; 0; 6; 5; 198.207.141.6;
ISS_SERV_IN; 824951202; 877494; 34; 78; 45; 0; 6; 5; 198.207.141.6;
ISS_START_READ; 824951202; 885279; 34; 78; 45; 0; 6; 5; 198.207.141.6;
ISS_END_READ; 824951202; 909439; 34; 78; 45; 0; 6; 5; 198.207.141.6;
ISS_START_WRITE; 824951202; 910743; 34; 78; 45; 0; 6; 5; 198.207.141.6; 49264
APP_RECEIVE; 824951210; 914210; 34; 78; 45; 0; 6; 5; 198.207.141.6;
APP_SENT; 824951202; 824949; 34; 76; 45; 0; 6; 3; 198.207.141.6;

Monitoring

- **NTP is used to synchronize time stamps throughout MAGIC**
  - **all MAGIC hosts run *xntpd*, which synchronizes the clocks of each host both to time servers and to each other**
  - **the MAGIC backbone segments are used to distribute NTP data, allowing us to synchronize the clocks of all hosts to within about 250 microseconds of each other, but...**
    - **many different sys admins (harder to synchronize than clocks)**
    - **the systems have to stay up for a significant length of time for the clocks to converge to 250 μs**

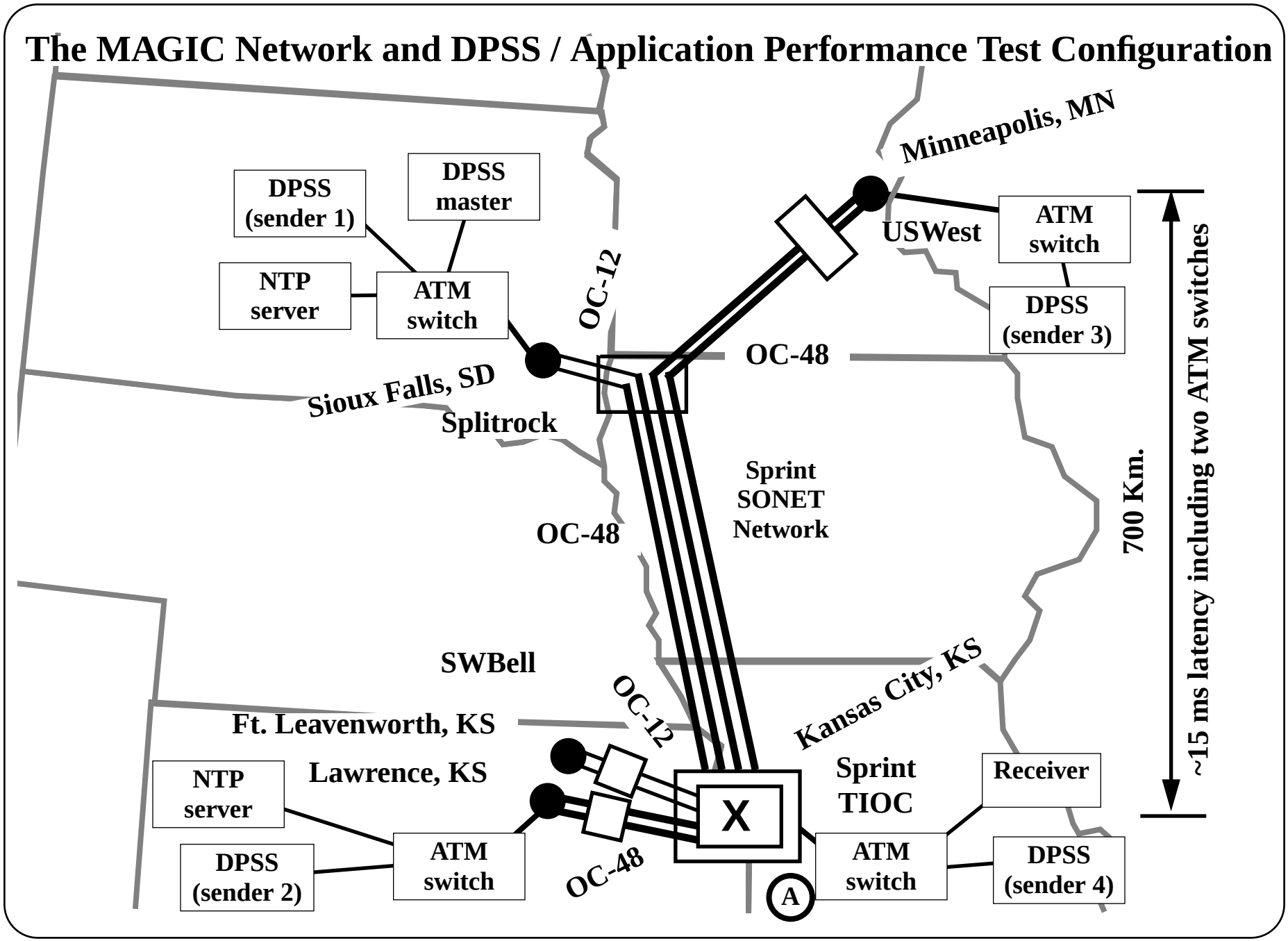
Monitoring


Long-term Time Convergence With NTP
"gpt.data.set1.avg"
NTP time vs. local GPS, difference in μs
1000
900
800
700
600
500
400
300
200
100
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Running time, hours
9am 10 11 12 1pm 2 3 4 5 6 7 8 9 10 11 12 1am 2 3 4 5 6 7 8

Monitoring Experiments

- **The monitoring experiments are carried out in the MAGIC environment:**
  - **a 700 Km diameter, star topology, configurable OC-48 SONET network**
  - **OC-3 and OC-12 ATM switches**
  - **backbone and “leaf” switches**
  - **dozens of directly attached hosts at six sites**

Monitoring Experiments



Monitoring Experiments

- **Network "tuning"**
  - **Initial studies by KU and MSCI established the optimal TCP parameters, and the cell pacing for the DEC Alphas (which, at the time, were the only senders capable of overrunning the switches)**
- **Server to Application Throughput (per server)**

DPSS Server Configuration	Burst (Mbits/sec)		Average (Mbits/sec)	
	TCP	UDP	TCP	UDP
Sun SS10-41; fddi	44	43	26	27
Sun SS10-42; fddi	62	56	37	40
Sun SS20-62; fddi	72	94	54	66
Sun SS20-62; atm	104	125	65	70
DEC Alpha; atm	134	-	72	90

Monitoring Experiments

- **End-to-End performance experiments**
  - **TerraVision (the “real” application) is run interactively**
  - **traces of data requests are collected**
  - **tv_sim is both a parameter and trace driven simulator (in trace mode, the application can be - and usually is - “speeded up”)**

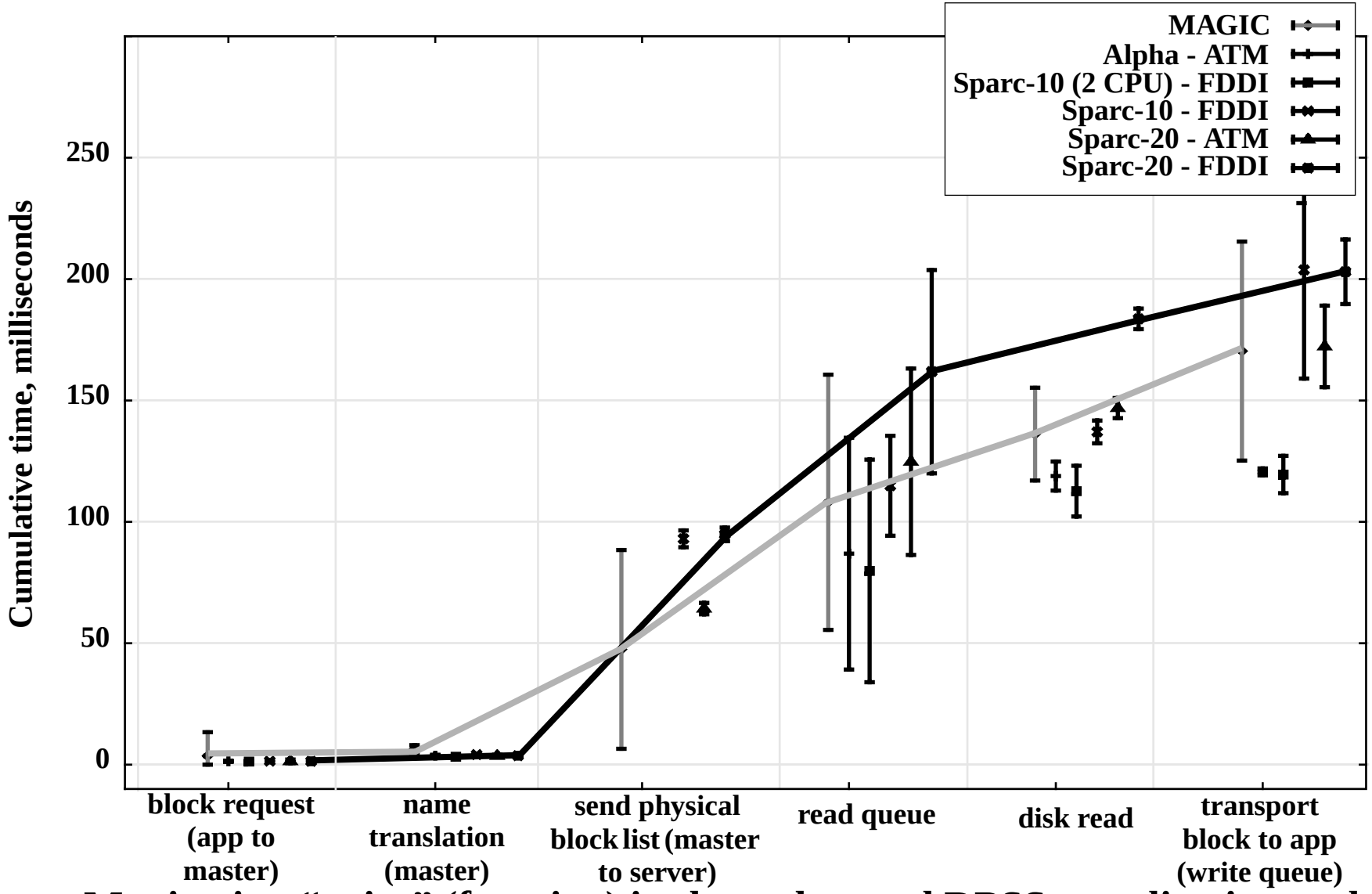
Monitoring Experiments

- **Overall system latency**
  - **with the current applications, we have found that the overall latency is primarily a function of the server platform configuration (e.g. how many disks) and the nature of the interconnection network**
  - **this can be demonstrated by just considering the monitoring information carried in the data units**

Monitoring Experiments

DPSS “Open Loop” Performance: Latencies in the architecture

(Latency at performance monitor points - “zero latency” application)

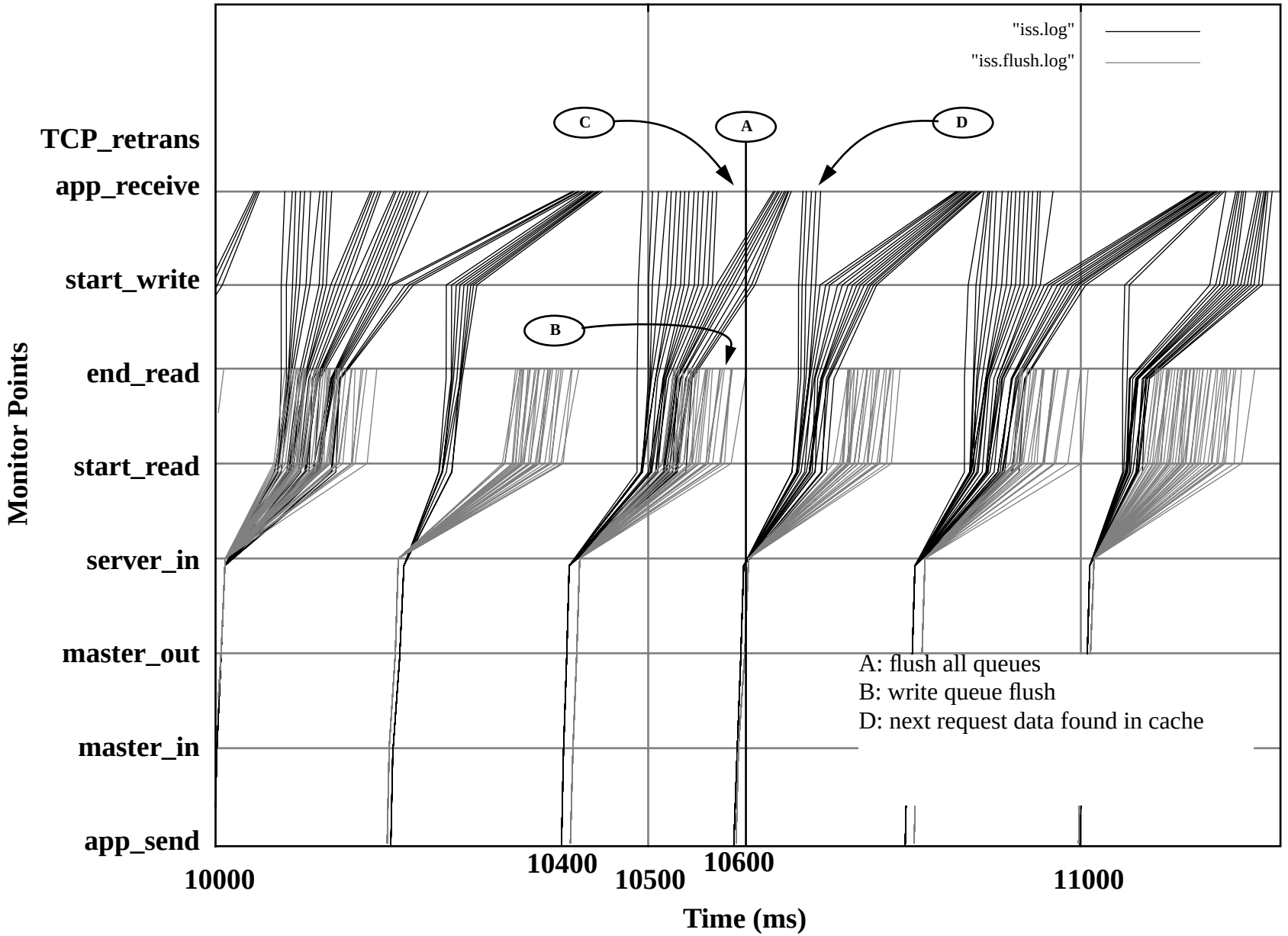


Monitoring “point” (function) in the end-to-end DPSS - application arch.

End-to-end, Top-to-Bottom Analysis

- **One server, LAN case**
  - **each line represents the history of a data block as it moves through the end-to-end path**
  - **data requests are sent from the application every 200 ms (the nearly vertical lines starting at *app_send* monitor point)**
  - **initial single lines fan out as the request lists are resolved into individual data blocks (*server_in*)**
  - **each block is first individually represented as requests in the read queue**
  - **the following figure illustrates:**

End-to-End, Top-to-Bottom Analysis



One server, ATM LAN, one SS-20 as server, tv_sim on DEC 3000/600

End-to-End, Top-to-Bottom Analysis

+ at Ⓐ flush of all queues due to next arriving block request list
+ at Ⓑ write queue flush because a transmission delay (at Ⓒ) prevented this collection of blocks from being transmitted before the next list arrived
+ at Ⓓ the blocks that were flushed from the write queue are retained in the memory cache, and the next request data found this group in cache

- lines terminate at *end read*, a few also end at *start read*
  If a data request is not satisfied before the next request list arrives, it is flushed (discarded), from all the queues, but not from the memory cache.

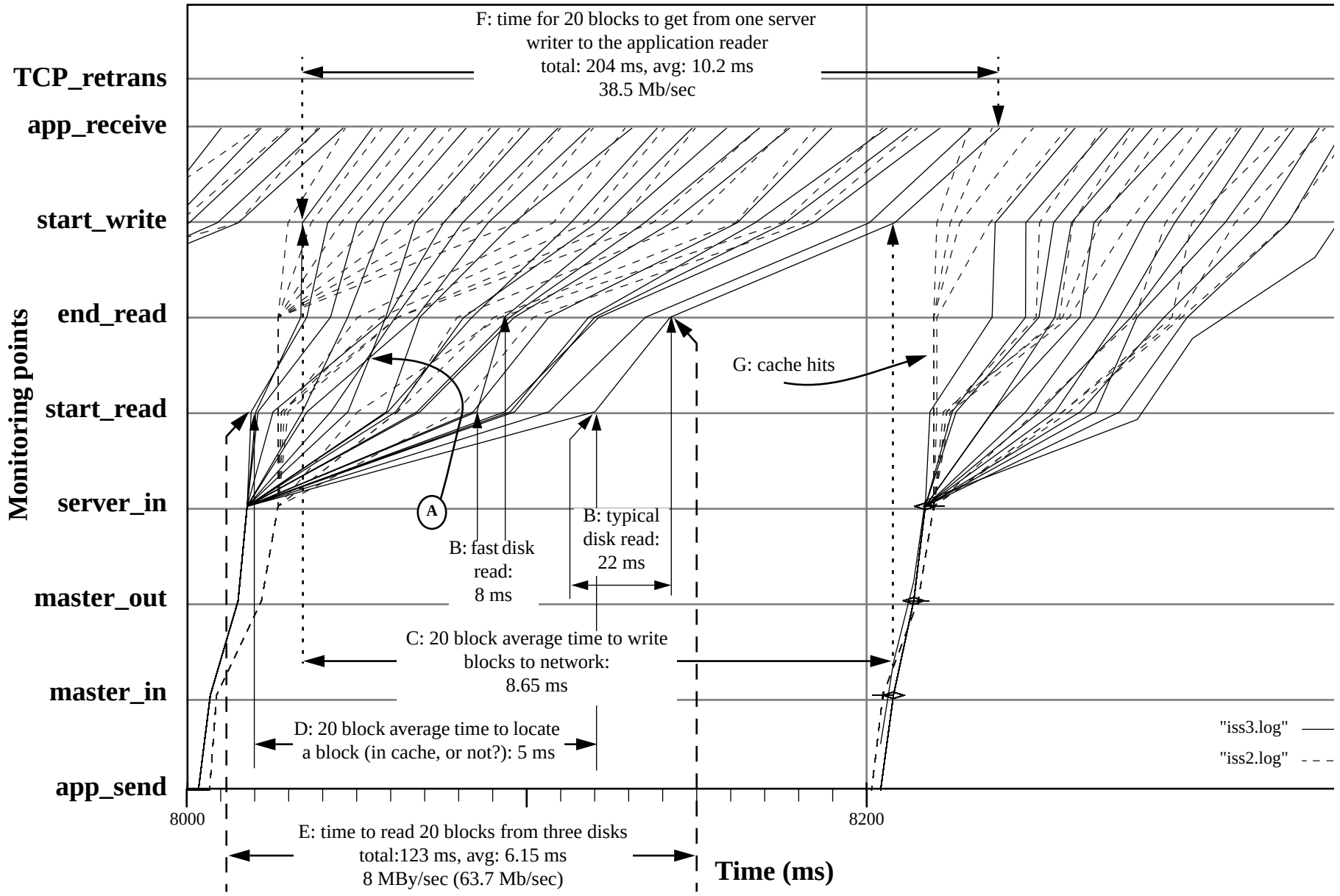
End-to-End, Top-to-Bottom Analysis

(Assumption is that a new data block request is more important than any unsatisfied previous requests, so discard all pending requests. The application needs to predict ahead. Even if requested data is not sent, it is cached in memory, where it will remain available for faster retrieval (for a short time). This approach ensures that the entire data pipeline stays full, and that disk server resources are never idle.)

End-to-End, Top-to-Bottom Analysis

- **A two server LAN experiment:**
  - **detailed analysis of the data block life-lines (following figure) shows:**
    - **if two lines cross in the area between *start read* and *end read*, e.g. at Ⓐ, this indicates a read from one disk was faster that a read from another disk**
    - **G: some requested data are found in the cache (The two nearly vertical life-lines on the left of groups one and three show near zero queue residency times.)**
    - **B: since all the disks are the same type, the two characteristic read times are probably due the layout algorithm clustering data on a disk**

End-to-End, Top-to-Bottom Analysis



Two server, ATM LAN

End-to-End, Top-to-Bottom Analysis

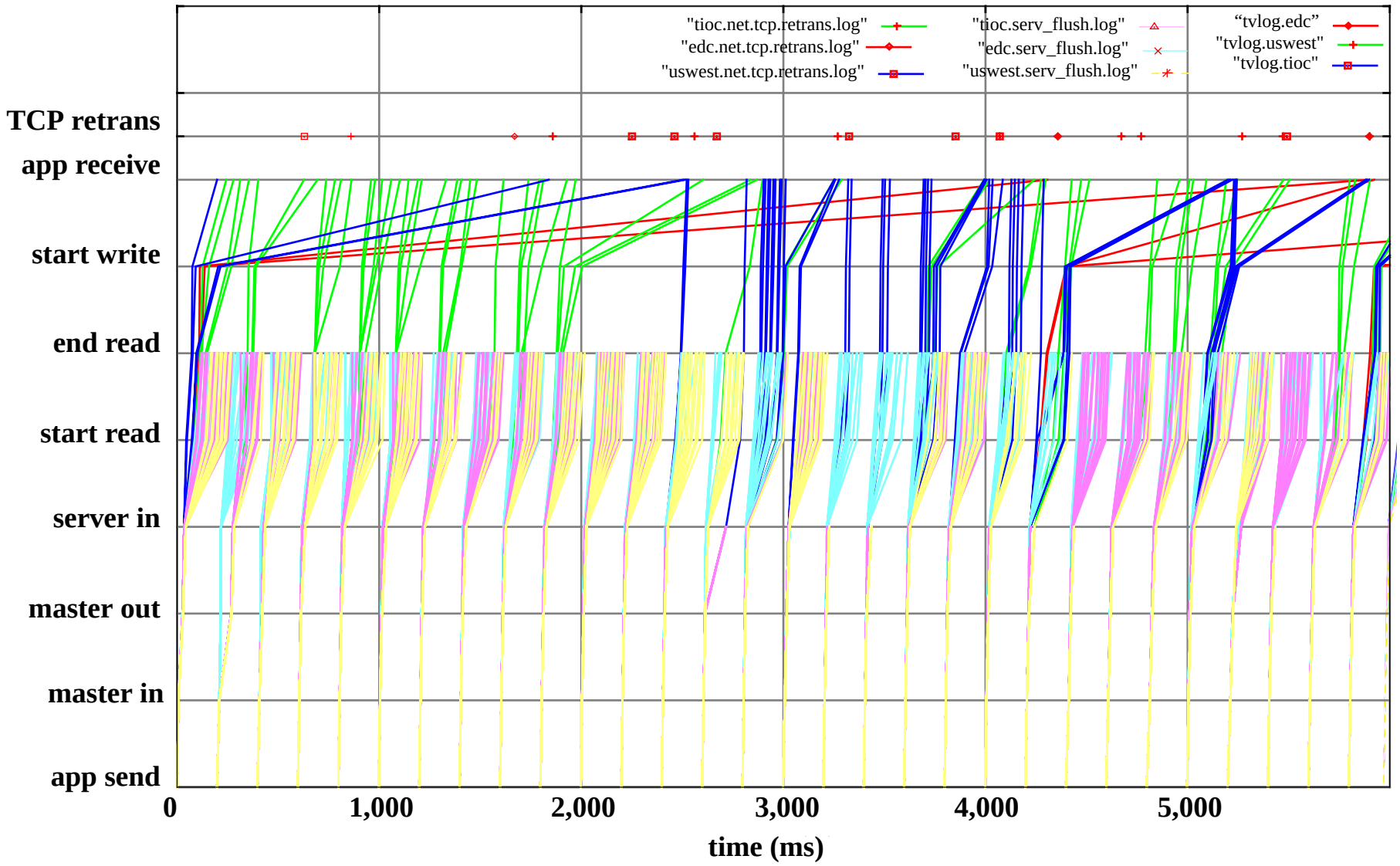
- **C: average time to move data from the memory cache into the network interface is 8.65 ms**
- **D: upon the receipt of a list of data blocks to locate and send, the average time to locate the block in memory cache or on disk is 5 ms**
- **E: the average read rate from four disks is 8 MBy/sec**
- **F: the average send rate (receiver limited, in this case) is 38.5 Mb/sec.**

End-to-End, Top-to-Bottom Analysis

- **Three server, WAN case (following figure)**
  - **TCP retransmissions and some very long delays (up to 5500 ms)**

    (Once a block is written to the TCP socket, the user level flushes have no effect, and TCP will re-send the block until transmission is successful, even though the data is likely no longer needed and is holding up newer data.)

End-to-End, Top-to-Bottom Analysis

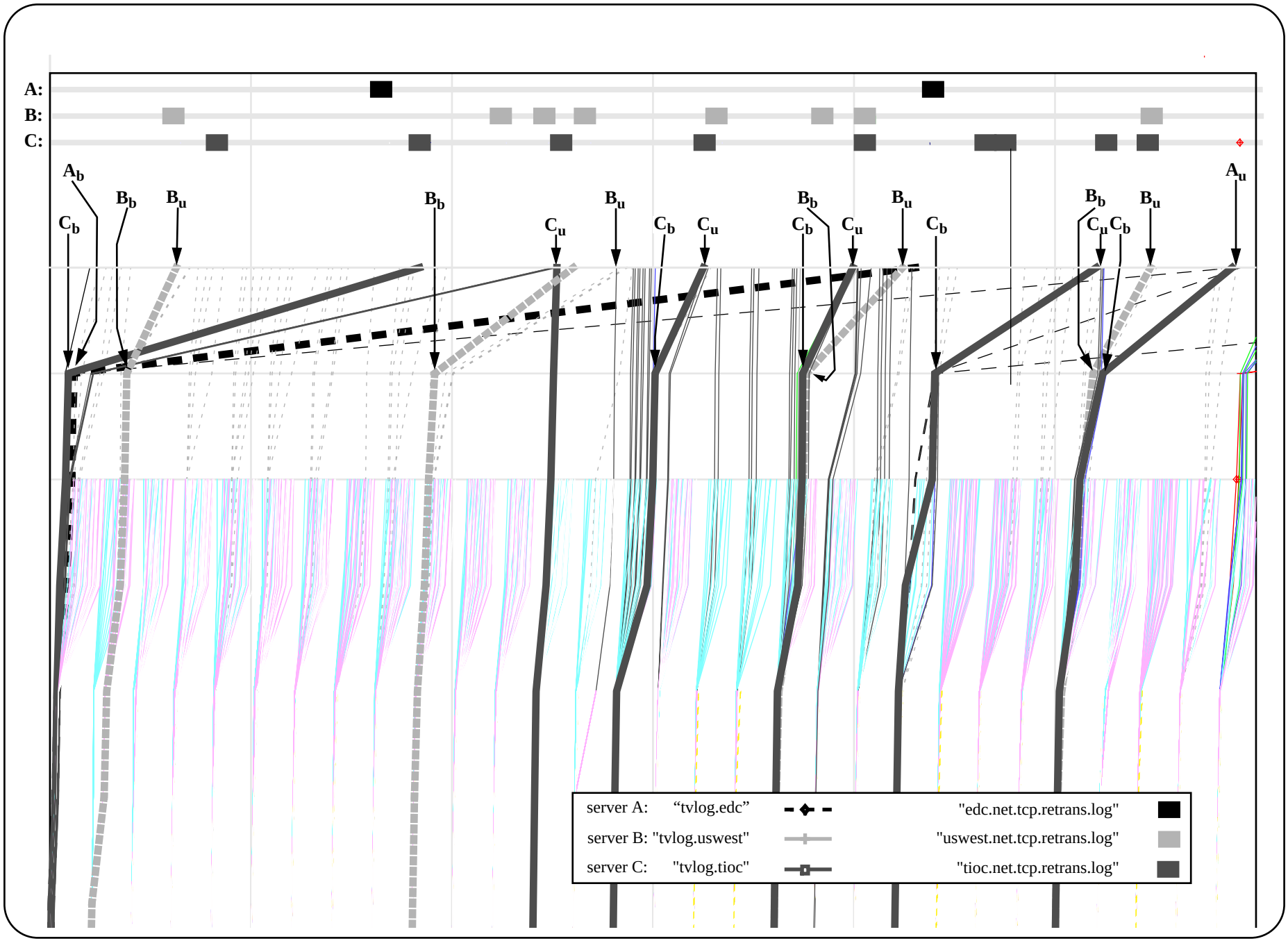


Three servers, ATM WAN, SS-10s as servers, tv_sim on SGI Onyx

End-to-End, Top-to-Bottom Analysis

- **First, let us analyze what the performance monitoring shows directly (the following figure shows some detail)**
  - **The long-delayed block life-lines (emphasized in this figure) illustrate the characteristic behavior of a data block getting into the write queue (*start_write* monitor point) and then incurring some very long delays getting to the application.**
  - **These long delays are almost always accompanied by one or more TCP retransmit events.**
    (The elongated blocks at the top of the figure indicate the interval during which the retransmit took place)

End-to-End, Top-to-Bottom Analysis



End-to-End, Top-to-Bottom Analysis

- **The start of the long delay transmissions are identified as A_b, B_b, or C_b (servers A, B, and C, in a blocked state).**

  The reason that the server is blocked as a whole (actually just one application is blocked since each application has its own TCP connection to the disk server) is that once a data block is written to the TCP socket, the user level flushes have no effect, and TCP will re-send the block until transmission is successful, even though the data is likely no longer needed and is holding up newer data.

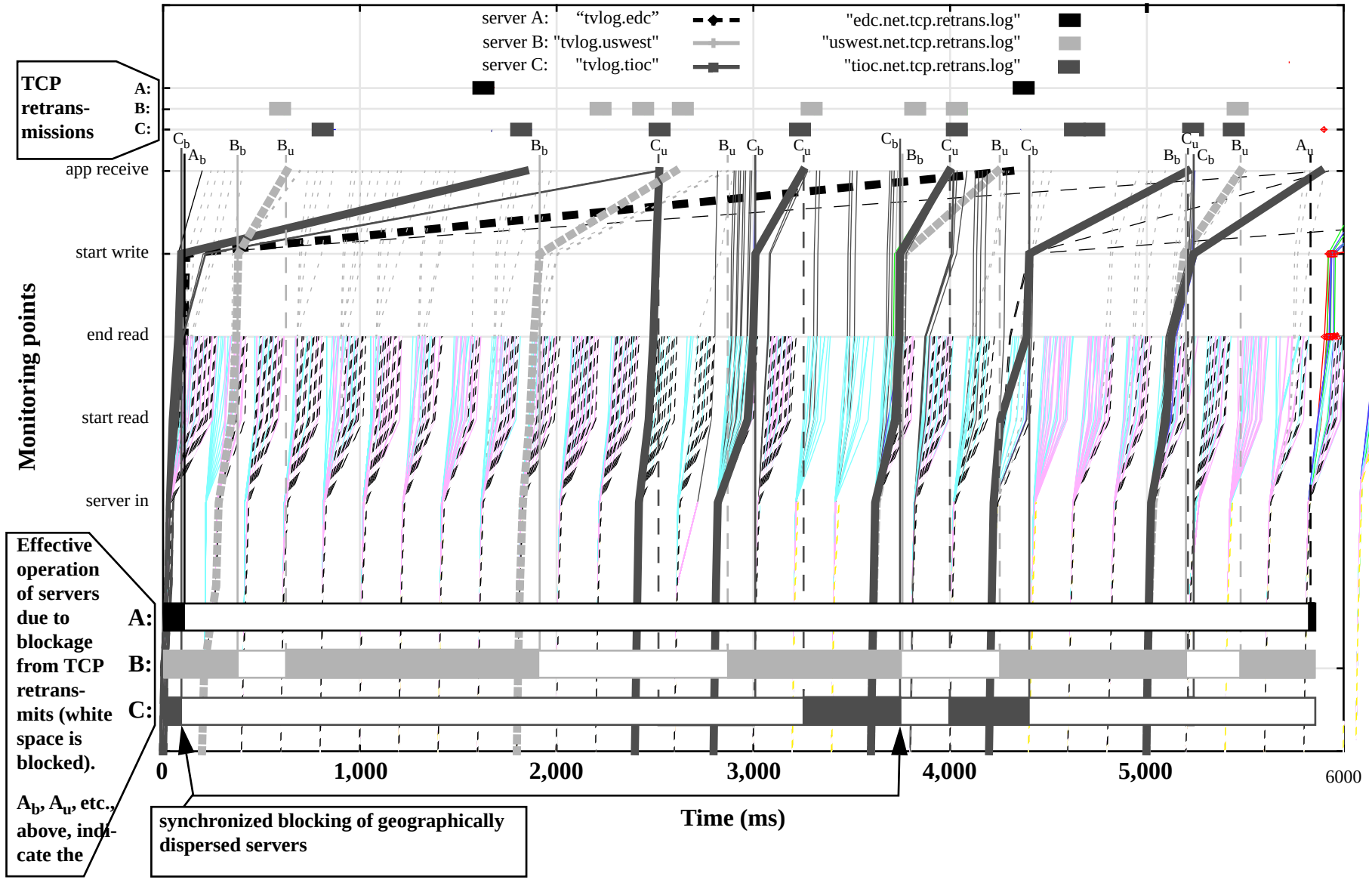
- **The server unblocks when the a retransmission is successful, letting the next write proceed. These unblock "events" are labeled A_u, B_u, and C_u.**

End-to-End, Top-to-Bottom Analysis

- **The impact of this behavior on the server as a supplier of data is shown at the bottom of the next figure**
  - **the horizontal bars indicate normal server operation by gray regions, and the blocked state by white regions**
  - **these server operation profiles show almost unbelievably poor performance: With three servers operating (EDC, USW, TIOC) we obtained throughput of 0.240 Mbits/sec, 3.8 Mb/s, and 4.4 Mb/s respectively, to deliver 106 blocks in 5 seconds out of about 900 requested blocks**

  This is in a network with minimum link speeds of 100 Mb/s and servers each with a verified data delivery capability of at least 45 Mb/s.

End-to-End, Top-to-Bottom Analysis



End-to-End, Top-to-Bottom Analysis

- **Beyond the poor performance, the operation profile clearly shows several places where there seems to be synchronization among the long-delay intervals that probably indicates the cell-interleaved TCP streams problem reported by Romanow and Floyd.**

  If so, an implementation of the Romanow and Floyd, Early Packet Discard (EPD) algorithm in the switches might help. (At the time of the experiments reported here, EPD was not implemented in the MAGIC ATM switches, but should be by the Spring of 1996, and we will redo the experiments to see if EPD helps.).

End-to-End, Top-to-Bottom Analysis

- **What we believe to be happening in this experiment is that TCP's normal ability to accommodate congestion is being defeated by an unreasonable network configuration:**
  - **the final ATM switch (at Ⓐ in the "Test Configuration" figure) is where the three server streams come together, and this switch has a per port output buffer of only about 13K bytes.**
  - **the network MTU (minimum transmission unit) is 9180 Bytes (as is typical for ATM networks)**

End-to-End, Top-to-Bottom Analysis

- **the TCP congestion window cannot get smaller than the MTU, and therefore TCP's throttle-back strategy is pretty well defeated: On average, every retransmit fails, even at TCP's "lowest throughput" setting, because this smallest unit of data is still too large for the network buffers.**
- **So, the situation is that three sets of 9 KBy IP packets are converging on a link with less than 50% that amount of buffering available, resulting in most of the packets (roughly 65%) being destroyed by cell loss at the switch output port.**

End-to-End, Top-to-Bottom Analysis

- **Acceptable Solutions and Unacceptable Solutions**
  - **TCP's normal ability to adapt to congestion should be able to make better use of this network, even with the mis-configured switch. The problem is that TCP is being prevented from providing an effective response to this congestion because it cannot reduce the congestion window to a small enough size.**
  - **TCP's adaptation mechanisms normally accommodate mapping a collection of high offered bandwidth streams (e.g. from the DPSS servers) into a low bandwidth path. The problem revealed in our experiments is that the TCP adaptation mechanisms make certain assumptions about the network that are not true for the MAGIC network as configured during our experiments.**

End-to-End, Top-to-Bottom Analysis

- **The important assumption for the issue at hand is that the amount of output port buffering needs to be of at least the order of Number_of_streams × TCP_min_retransmission_size. In other words, there needs to be at least enough buffering to make the lowest-throughput TCP retry efforts successful.**
- **This is the assumption that is violated in the current MAGIC configuration environment in which our prototype production experiment performed so badly. If the output ports of a switch or router are not large enough to accommodate (at least several) minimum retransmission units, then it is clear that retransmissions, even at the lowest throughput that the TCP adaptation mechanism can operate at, will fail, and TCP will therefore fail to provide optimal, or even reasonable, use of link capacity.**

End-to-End, Top-to-Bottom Analysis

- **With the current switch configuration, the TCP congestion window size should probably be of the order of 256 Bytes in order to accommodate three streams.**

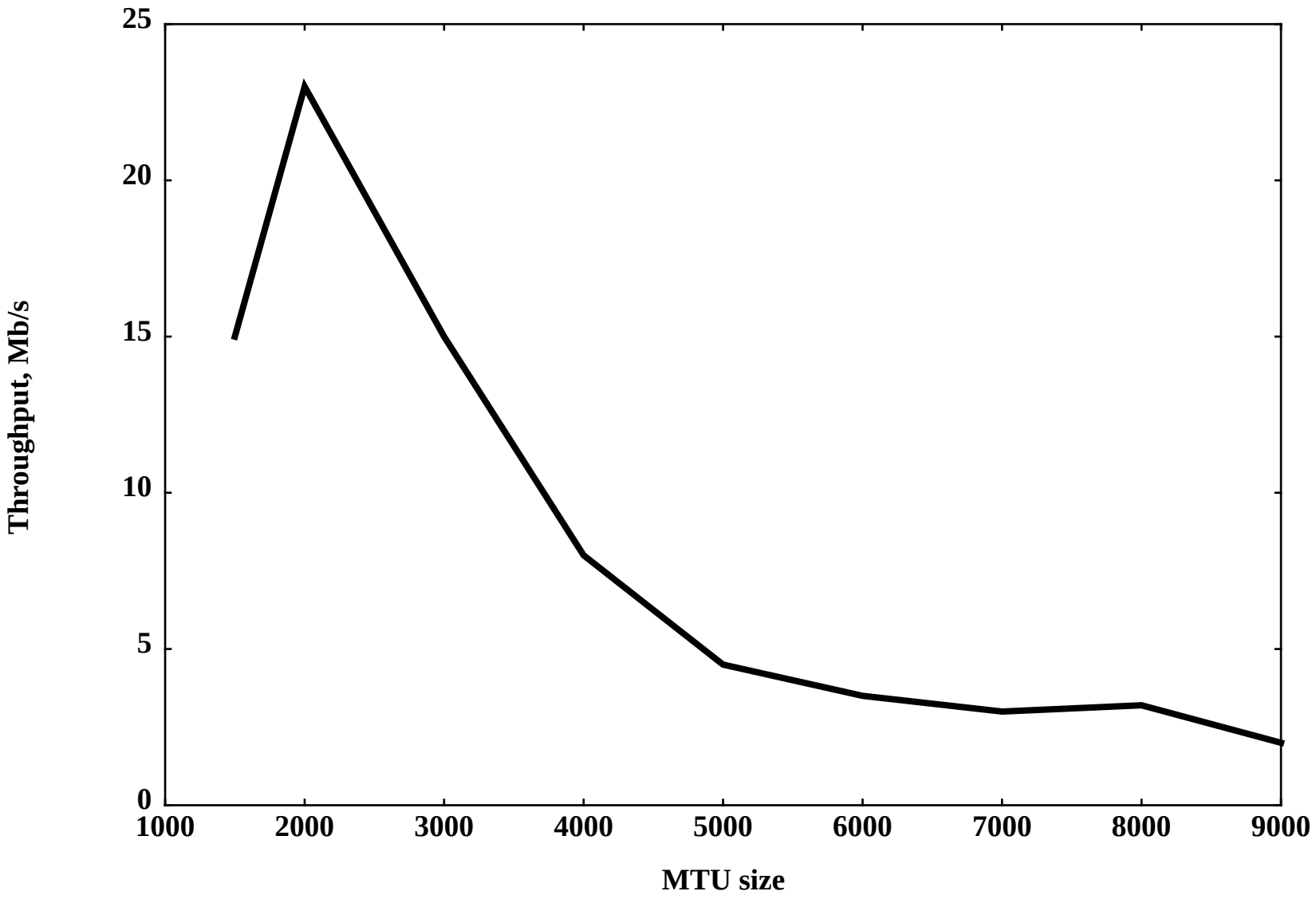
End-to-End, Top-to-Bottom Analysis

- **Apart from re-engineering the network - which may or may not be possible in general - or modifying TCP (which we will do in MAGIC and rerun the experiments) what else can be done?**
  - **ATM Congestion Control: There are various ATM cell-level congestion control schemes being developed by the ATM Forum Traffic Management Working Group that might help alleviate this problem. The ATM Forum Available Bit Rate (ABR) service specification includes a mechanism for switches and destinations hosts to send congestion and flow control information back to the source host congestion is detected. The first version of this should be available in UNI 4.0, which vendors will probably start shipping by the end of this year.**

End-to-End, Top-to-Bottom Analysis

- **Small MTU Size: We can reduce the network MTU to a very small value (e.g. 256 bytes). This should demonstrate the ability of TCP to deal with this type of congestion, and we expect that this approach will ameliorate the situation described above.**
  - **These experiments will be run in the near future, and we will post the results to http://www-itg.lbl.gov/DPSS/Experiments.)**
  - **Experiments done at the University of Kansas, Lawrence show that both large MTU size (illustrating the problem noted above) and small MTU size (preventing high performance operation of the servers) lead to low throughput. Data from the KU experiments done in the MAGIC testbed) indicates the relationship between TCP throughput with varying MTU size. (next figure)**

End-to-End, Top-to-Bottom Analysis



Single Stream Throughput vs. MTU Size for TCP Using Large Windows in an ATM WAN

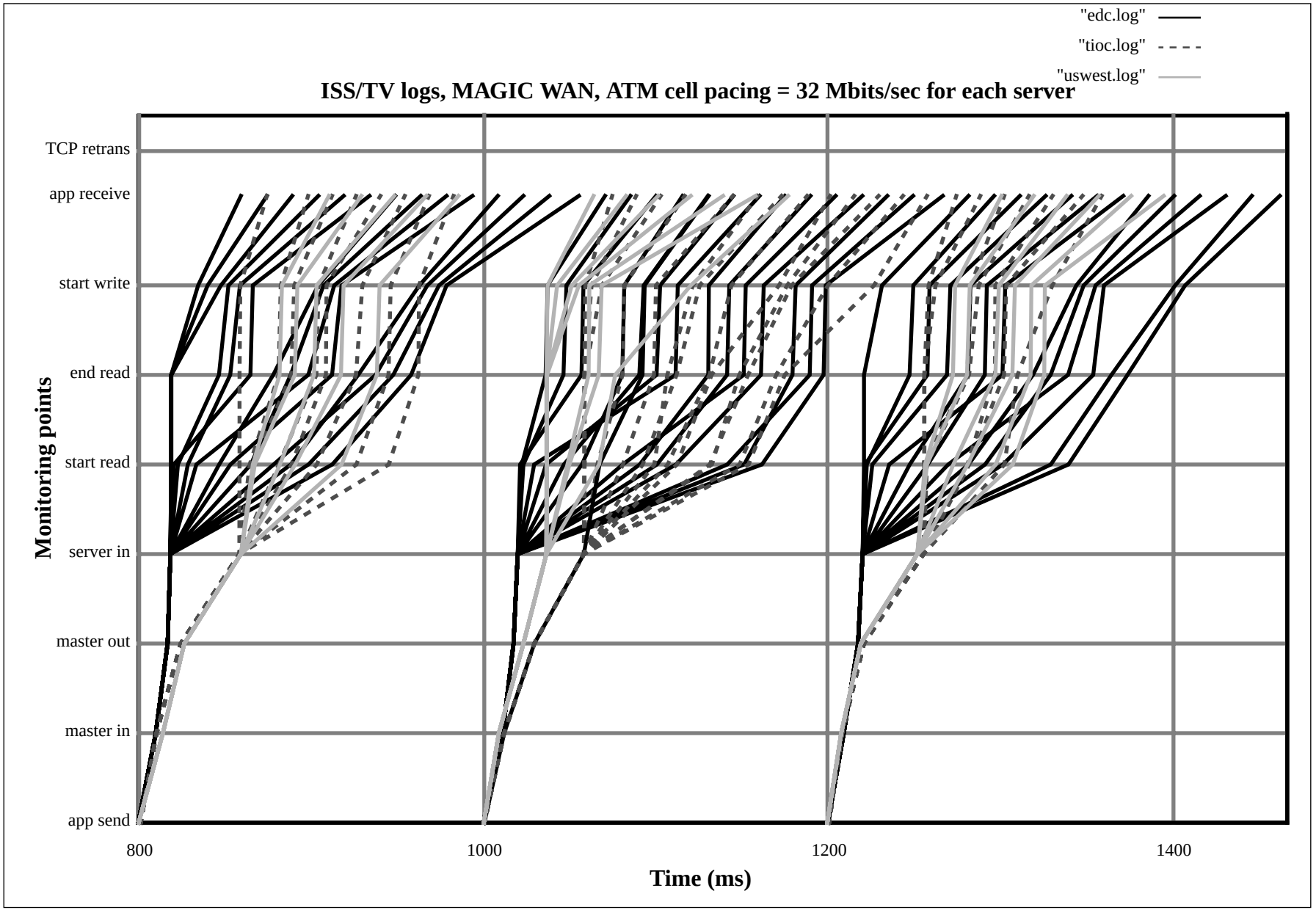
End-to-End, Top-to-Bottom Analysis

- **This approach, however, is clearly not an acceptable general solution because it prevents high performance operation of distributed applications everywhere in the network.**

End-to-End, Top-to-Bottom Analysis

- **Cell Pacing: We can also cell pace the disk servers to 1/N of the "broken link" bandwidth.**
  - **Cell pacing is bandwidth limiting at the level of ATM cells.**
  - **The cell pacing experiment has been done: we have cell-paced all of the DPSS servers at 1/3 the final link capacity so that we know that the total offered load does not exceed the output link capacity of the final switch. Not surprisingly, this approach corrects all of the observed anomalies. Under these conditions, all three servers provided roughly equal throughput of 30 Mb/s, and the delivery of blocks was "well behaved". (see next figure)**
  - **However, from our point of view, this again is not an acceptable solution.**

End-to-End, Top-to-Bottom Analysis



The Three Server ATM WAN Experiment with Cell Pacing

End-to-End, Top-to-Bottom Analysis

It solves the problem by reducing everything to the lowest common denominator. It also assumes that you will know a priori how many streams will be coming a given link. As a "lowest common denominator" approach, this does not allow individual servers to use available bandwidth when, e.g., other servers are not transmitting because the data happens not to be evenly distributed; when a server or network link fails; etc.

End-to-End, Top-to-Bottom Analysis

- **Re-engineering the Network: The ATM LAN experiments reported above were done using a Fore Systems ATM switch with rev. C network modules. These modules (set of four output ports and line drivers) have large output port buffers, and this appears to solve the congestion problem that we have been describing. These new switch output modules have 624 KBytes of buffering per port, so assuming minimum TCP segment sizes of 9180 Bytes (the ATM MTU), the switch module should be able to support up to 69 simultaneous maximally throttled TCP connections.**

  In the case of the MAGIC testbed, we can and are, re-engineering the network. In this case, the small buffer switch in the final link will be upgraded to provide much larger output port buffering. Once this is done, we will run a range of

End-to-End, Top-to-Bottom Analysis

experiments, including varying the switch output port buffer size, turning “early packet discard” off and on, etc. to verify that the various changes produce the expected results. Again, these experiments will be posted to the URL indicated above.

End-to-End, Top-to-Bottom Analysis

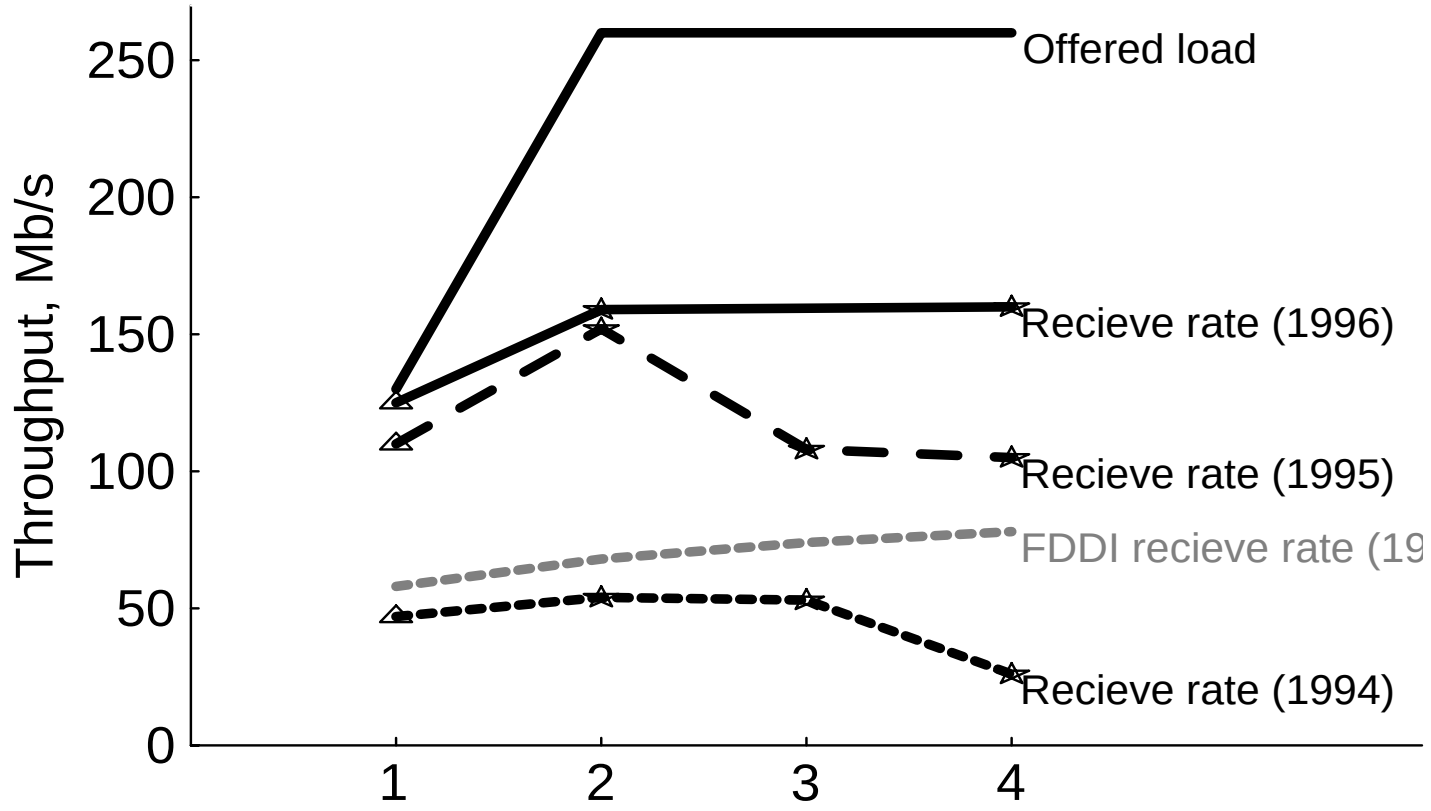
- **The UDP (or RTP) transport model does not suffer from TCP's blind retransmission problem because a single block is never retransmitted by the disk server - it is re-requested by the application if it is still needed the next time around**
  - **However, we have focused on TCP because:**
    - **We have other TCP applications that we need to have work properly.**
    - **In order for RTP to make a "reasonable", general purpose transport protocol in these circumstances, we have to implement most of TCP's congestion response mechanisms in our user-level code. This is planned, but not yet done.**

The Evolution of the IP over ATM Infrastructure

- **1994**
  - **single interfaces were slow**
  - **multiple interfaces were no faster (poor independence of data paths - nothing was multi-threaded)**
  - **switches dropped cells (silently)**
- **1995**
  - **single interfaces are faster**
  - **multiple interface data path independence is improving**
  - **switches still drop cells (silently)**

Evolution of IP Over ATM

Performance of Multiple ATM Interfaces
Receiving Multiple Data Streams on a Single System



Number of simultaneous streams into one or two interfaces
(triangle = 1 interface case, stars = 2 interface case)

Evolution of IP Over ATM

- **1996**
  - **single interfaces and systems are approaching wire speed**
  - **multiple interfaces better still, but not ideal**
  - **switches do report cell loss**
  - **cell pacing is now critical for all platforms (due to speed of single systems)**

References

- Cavanaugh, John, Timothy Salo, "Internetworking with ATM WANs", (available as http://www.msci.magic.net/docs/magic/ip-atm.ps)
- Evans, Joseph B., Victor S. Frost, Gary J. Minden, "TCP and ATM in Wide Area Networks", CNRI Gigabit Network Workshop '94. (http://www.magic.net/tcp/overview.html)
- Ewy, B. J., J.B. Evans, G.J. Minden, and V. S. Frost, "TCP/ATM Experiences in the MAGIC Testbed", Fourth IEEE Symposium of High Performance Distributed Computing, August 1995, pp. 87-93.
- Jonkman, Roelof J.T., "An Overview of NetSpec", Telecommunications & Information Sciences Laboratory, University of Kansas. (http://www.tisl.ukans.edu/Projects/AAI/products/netspec/)
- Lau, S, and Y. Leclerc, "TerraVision: a Terrain Visualization System,", Technical Note 540, SRI International, Menlo Park, CA, Mar. 1994. Also see: http://www.ai.sri.com/~magic/terravision.html
- Richer, I. and B. B. Fuller, "An Overview of the MAGIC Project," M93B0000173, The MITRE Corp., Bedford, MA, 1 Dec. 1993. (Available from http://www.magic.net/MAGIC_Summary.ps.)
- Romanow, A., and Floyd, S., "Dynamics of TCP Traffic over ATM Networks." IEEE JSAC, V. 13 N. 4, May 1995, p. 633-641. (An earlier version appeared in SIGCOMM '94, August 1994, pp. 79-88.) See http://ftp.ee.lbl.gov/floyd/epd.html

- **Tierney, B., Johnston, W., Chen, L.T., Herzog, H., Hoo, G., Jin, G., Lee, J., "Using High Speed Networks to Enable Distributed Parallel Image Server Systems", Proceedings of Supercomputing '94, Nov. 1994, LBL-35437. Available from http://www-itg.lbl.gov/ISS/papers.html.)**

- **For more information see**

  http://www-itg.lbl.gov/DPSS

  (much of the material in this presentation is described in more detail in a draft of a paper to appear in IEEE Networking)

  and

  http://www.magic.net

  on the World Wide Web.